Utilizing Massive Spatiotemporal Samples for Efficient and Accurate Trajectory Prediction

Addison Chan and Frederick W.B. Li

Abstract—Trajectory prediction is widespread in mobile computing, and helps support wireless network operation, location-based services, and applications in pervasive computing. However, most prediction methods are based on very coarse geometric information such as visited base transceiver stations, which cover tens of kilometers. These approaches undermine the prediction accuracy, and thus restrict the variety of application. Recently, due to the advance and dissemination of mobile positioning technology, accurate location tracking has become prevalent. The prediction methods based on precise spatiotemporal information are then possible. Although the prediction accuracy can be raised, a massive amount of data gets involved, which is undoubtedly a huge impact on network bandwidth usage. Therefore, employing fine spatiotemporal information in an accurate prediction must be efficient. However, this problem is not addressed in many prediction methods. Consequently, this paper proposes a novel prediction framework that utilizes massive spatiotemporal samples efficiently. This is achieved by identifying and extracting the information that is beneficial to accurate prediction from the samples. The proposed prediction framework circumvents high bandwidth consumption while maintaining high accuracy and being feasible. The experiments in this study examine the performance of the proposed prediction framework. The results show that it outperforms other popular approaches.

Index Terms—Location-dependent and sensitive, pervasive computing, location-based services, trajectory prediction

1 Introduction

Predicting the trajectories of mobile users accurately is beneficial to pervasive computing and locations-based services (*LBSs*) because appropriate information can be processed and fetched to subscribers in advance. This alleviates system latency and increases the interactivity of services. Other applications, such as, early reminder system [1], traffic planning [2], and dead reckoning in GPS [3], also require accurate prediction of user locations. Trajectory prediction also supports the operation of mobile networks, for example, bandwidth management [4], location management [5], and routing in mobile ad hoc networks (*MANETs*) [6].

Most prediction methods in mobile computing model a future trajectory as a sequence of cells, location area (*LA*), or significant locations (e.g., school and shopping mall), which usually have a diameter of several kilometers. These methods can merely provide a coarse trajectory prediction. Therefore, employing very rough locations in prediction undermines prediction accuracy, which in turn restricts the methods' variety of applications. Some studies exploit precise spatiotemporal samples for more accurate prediction, using recently emerging positioning technology such as GPS and femtocells, whose error is typically less than 50 meters.

However, to apply precise spatiotemporal samples, we need more bytes to represent the samples. Thus, the data size of those spatiotemporal samples is often massive. The samples can be transmitted to and from a predictor, which is located at mobile stations (*MS*) or mobile operator networks. When these samples are transmitted via networks, a large amount of bandwidth usage is introduced. This bandwidth issue discourages the use of precise spatiotemporal samples for prediction, and causes the dilemma of accurate prediction and minimal bandwidth usage. Some related studies use precise spatiotemporal samples, but they limit the usage to determining the parameters in a fixed, predefined motion model for predicting trajectory [7], estimating MS states for short-term trajectory prediction (on the order of milliseconds) [6], or constructing significant locations for coarse trajectory prediction [1]. Those studies cannot provide a prediction method for all kinds of moving characteristics and long-term forecasting (on the order of hours).

To employ precise spatiotemporal information, the availability of the information must be considered. There are several factors dominating the availability. The first factor is the accuracy of positioning technology. The primary positioning technology is GPS, whose error is on the order of few meters. Some techniques, such as in [3], [8], and [9], can maintain accurate positioning when GPS is absent. However, positioning incurs extra power consumption, and shortens mobile devices' battery life. Thus, the second factor is the power consumption of mobile devices. Many researches have designed low powered components for mobile devices, for example, in GPS receiver [10], CPU [11], display [12], and networking [13]. Apart from energy saving, there have been breakthroughs in battery technology that improve both charge capacity and recharging time by about 10 times, such as in [14]. The last factor is the popularity of mobile applications or LBSs, which encourage

mobile users to stay online. Some social networking applications with LBS supported, for example, Facebook and Google+, have been prevalent, and attracted hundreds of million users. Longer online duration results in higher availability of spatial-temporal information.

Concluding all of the three factors we have discussed, acquiring massive precise spatial-temporal samples for trajectory prediction is promising. This conclusion motivates us to propose a novel trajectory predictor that utilizes those samples from MSs. Given spatiotemporal samples for predictor training and the past trajectory of an MS, this proposed predictor can accurately predict the future trajectory of the MS without presuming any motion model. This predictor attains a higher accuracy than other methods using the same amount of information. Equivalently, it can achieve an accuracy level using less network bandwidth. To utilize massive precise spatiotemporal samples efficiently, the proposed predictor identifies and extracts the information that is beneficial to accurate prediction from the samples. Our proposed prediction approach and its novelties are summarized as follows:

1. The proposed approach quantizes MS trajectories both spatially and temporally to reduce computation, and the bandwidth usage due to transmitting spatiotemporal samples. The quantization is adaptive to the probability distribution of MS locations and the duration of staying at those locations so that important information for prediction is retained.
2. The information that does not significantly contribute to future movement is ignored. That information is obtained by learning which historical data are unimportant for prediction. The amount of data required to process is then greatly reduced while prediction accuracy is not sacrificed.
3. After capturing the relevant information for prediction, we perform a transformation so that sample trajectories with similar probabilistic characteristics are clustered, and an accurate and efficient prediction is facilitated. This is achieved by nonlinear *support vector machine* (*SVM*) [15].
4. To raise the reliability of the predictor, the proposed method considers multiple possible future locations and combines them into final prediction results. If the reliability of a predicted location is low, it is excluded from the prediction results to avoid prediction with huge error, and wastage of bandwidth due to sending out unreliable prediction results from the predictor.

We include extensive experiments showing that the proposed predictor outperforms other popular predictors in terms of accuracy and bandwidth usage.

This study focuses on exploiting the most essential information, i.e., past trajectories, for prediction. It is possible to take additional contexts into account to enhance prediction accuracy as in [16], for example, user favorites. However, it may be costly to define the contexts. Besides, the interpretation of the context may be subjective and vary with different people. The remainder of this paper is organized as follows: Section 2 reviews different types of prediction methods. Section 3 provides an overview of the proposed predictor while Section 4 describes the predictor in detail. The performance of the proposed predictor is evaluated, and compared with other popular prediction methods in Section 5. Finally, Section 6 concludes this paper.

2 Related Work

There are a lot of methods for trajectory or movement prediction. They can be found in many aspects in addition to mobile computing, such as virtual reality, augmented reality, computer vision, and robotics. We discuss them in this section. The prediction methods can be generally categorized into two approaches: 1) employing deterministic model, and 2) without deterministic model.

2.1 Prediction Employing Deterministic Model

One of the common approaches is modeling trajectory or movement by fixed and predefined models. Some types of modeling consider the underlying dynamics or kinematics of the movement, for example, polynomial predictors in [6] and [17]. Some studies apply Gauss-Markov process to model the movement [18], [19]. Noise filtering can be applied together with these models to increase prediction accuracy, for example, Kalman filter, unscented Kalman filter [20], and sequential Monte Carlo method [21]. Considering underlying dynamics can produce precise prediction. However, the prediction accuracy plummets when we predict a few hundred milliseconds ahead, and the movement being predicted is with high frequency (the movement is unsteady or arbitrary) [17], [22]. Trajectories can be also forecasted by some common models for time series data [7], for example, *ARMA*, *ARIMA*, and *ARFIMA*, weighted sum of the states of moving objects in the past [23], or an artificial neural network (*ANN*) [24]. Instead of modeling trajectory, some studies strive to model transition probabilities, which represent the probabilities of MSs transiting from one cell or region to another one. These probabilities are evaluated according to some predefined rules or formulae [25]. To exploit even more information, Samaan and Karmouch [16] take more contexts of mobile users and locations into account for better probability estimations, for example, the users' interests and purposes of the locations. Predictors employing fixed model are suitable for specific movement pattern or applications, for example, in [22] and [23].

2.2 Prediction without Deterministic Model

Neither fixed models for the movement trajectory nor transition probability can predict MS movements accurately if the models do not fit the movements. Therefore, researchers attempt to develop prediction methods or prediction methods without presuming any model. These methods are solely based on statistics or past trajectories. They apply the samples in the past to define an appropriate model implicitly. The methods are suitable for any kinds of movement, which exhibit different characteristics due to unlike underlying dynamics or kinematics.

One of the methods without presuming any movement model is Markov chain modeling [26]. Most studies apply first-order Markov chain [5], [27] or second-order one [1], [4], [28]. It is also possible to estimate the cell residence time

by amending Markov chain to be Markov renewal process [29]. Markov chain of very few orders may not be able to capture all characteristics of user movements for prediction. Thus, some papers suggest the prediction methodologies in compression, such as Lempel-Ziv-78 (*LZ78*) [30], which need not limit the order of Markov chain, and therefore achieves very high-order Markov chain modeling. These methodologies were originally designed for retaining information in lossless compression and not for movement prediction. However, the higher order of Markov chain we adopt, the more historic data a predictor requires. Including more data implies a higher consumption of bandwidth and computation resources. This consumption may be wasteful because Markov chain with higher order does not guarantee a more accurate trajectory prediction [31]. Another issue for Markov modeling is the definition of discrete cells. Most studies define the cells according to the location and coverage of BTSs or access points (*APs*) [5], [26], [30], [31]. Some papers define the cells as regular square grids [28] or as meaningful locations, for example, campus and home [1]. Ping-pong phenomenon [27] and road topology [4] can be also considered in defining cells. Unfortunately, quantizing positions into discrete cells, which are usually very large, loses precise information for prediction. The cells can be made smaller, but it results in an excessive number of cells, and leads to huge computation. Also, more bytes are required to represent a cell, and more number of position samples is needed to represent a trajectory. The data size involved in prediction increases significantly. The bandwidth usage thus becomes vast.

Trajectory similarity matching is another approach past trajectories to predict mobile user movements. This approach compares the current trajectory of mobile users with the sample trajectories in the database that collects MS movement records. The sample trajectory that is the most similar to the current trajectory is then matched and retrieved for prediction. Since the whole trajectory is examined, this approach has the advantage of high-order Markov chain modeling, which examines a long history. In mobile computing, the trajectories are usually in form of discrete cells sequences. Thus, the trajectories can be treated as strings, and the matching can be achieved by comparing the strings [32]. The matching can be also based on trajectory characteristics, such as length, duration, and frequency of occurrence [33]. As discrete cells comprise the trajectories, decision tree learning can be applied to trajectory prediction [34]. To reduce computation, similar sequences are clustered and merged into one sequence or trajectory [32]. However, representing a trajectory as a discrete cells sequence for matching ignores some spatial information, which also happens in the Markov chain modeling. This representation does not include the information about the spatial relationship among discrete cells. Some representation is in form of a sequence of exact position coordinates [35], [36]. Like clustering the sequences of discrete cells, the sequences of position coordinates can be clustered. It is accomplished by a clustering algorithm, for example, K-means. The trajectory matching becomes comparison between the current trajectory and the centroid of each cluster. Similar to high-order Markov chain, trajectory similarity matching approach may waste computation and bandwidth due to examining too much history. This approach also assumes that two trajectories will exhibit similar future outcomes if their overall geometrical similarity is high. This is not true if each part of the trajectories contributes to future MS behaviors differently. Furthermore, this approach treats the whole retrieved trajectory as the predicted trajectory. This ignores the fact that, in general, the prediction error along the predicted trajectory is uneven. Unreliable prediction may ensue from some portions of the predicted trajectory.

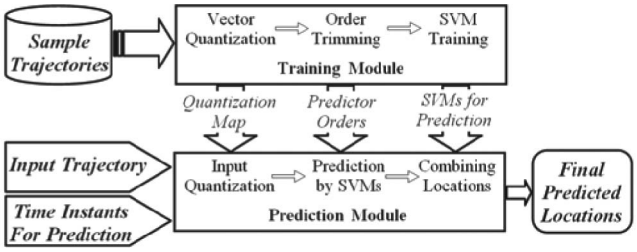

Fig. 1. The prediction framework overview.

3 The Proposed Prediction Framework

The proposed prediction framework predicts MS locations based on the information from sample trajectories captured from past MS trajectories. Fig. 1 shows the overview of the framework. The proposed framework is separated into training and prediction modules. The sample trajectories and the input trajectory are in form of a sequence of n^* positions or locations. Those trajectories are represented as $\{l^*_{n^*}, l^*_{n^*-1}, \ldots, l^*_1\}$ initially, where $l^*_{n^*}$ is the latest location. The locations are sampled regularly, for example, every 100 seconds. The training module extracts the information from the sample trajectories from which we can infer the future trajectories. The extracted information is then applied to the online prediction in the framework. The online prediction, which is executed in the prediction module, takes the input trajectory and the time instants for prediction at when the predicted locations are, for example, for every 100 seconds afterward. The input trajectory is the current trajectory of an MS. By applying the information from the training module, predicted locations at the specified time instants are evaluated.

The training and prediction module can reside at either an MS (client-based prediction, e.g., tablet PC), or mobile operator networks (server-based prediction, e.g., an operator's application server). If we perform the prediction at mobile operator networks, the current MS trajectories for online prediction and the sample trajectories for training are sent from the MS to the networks. It is because all trajectory samples are obtained from the MS. Network bandwidth is then consumed. We call this kind of bandwidth, which is responsible for receiving data for the training and prediction module, as *incoming bandwidth*. If we perform the prediction at MS, the predicted locations are sent out to mobile operator networks. Bandwidth is then used by sending out prediction results from the prediction module. We call this kind of bandwidth as *outgoing bandwidth*. Employing precise spatiotemporal samples causes a large

amount of data, which can overwhelm both incoming and outgoing bandwidth. This bandwidth issue is considered in our proposed framework.

The training module first takes the sample trajectories to conduct quantization, which is designed to partition an environment into cells. The environment is an area covered by mobile services. This quantization is not based on the coverage of BTS or AP. More popular regions should contain more tiny cells. This adaptive approach reduces the data size for representing a trajectory. Computation resource and bandwidth usage is thus lowered while important trajectory information is retained. The residence duration of each cell, which is the duration of staying in a cell, is also quantized adaptively according to how long MSs usually stay there. The quantization is performed by *vector quantization* (*VQ*) described in Section 4.1.

The quantized samples are then transformed into sequences of states. The next step in training is to determine how long the history or how many past states should be examined for the prediction. However, this decision depends on the current situation or state of the MS. For example, if the MS travels fast and straight, it will likely continue traveling straight, regardless of how it has traveled previously. On the other hand, if the MS is currently at a junction point, more information will be required to predict the next cell to be visited. More states of MS in the past will be examined in addition to the current state to determine the length of the history for prediction. This problem is equivalent to deciding the most suitable order in Markov chain modeling, which is called *predictor orders*. A high order demands heavy computation and bandwidth usage. Thus, the proposed method trims the order, while maintaining accurate prediction. This process is called *order trimming*, as Section 4.2 describes.

Order trimming greatly reduces the sample trajectories' sizes. The trimmed samples are then used in SVM training. The SVMs are classifiers that can alleviate the overfitting problem, and evaluate the class membership probabilities [37], [38]. The proposed predictor employs the SVMs to estimate the conditional probabilities of all possible output states, given the states in the past. Prior to SVM training, the input space of the samples is transformed into a desirable feature space that separates the samples demonstrating different future outcomes, and facilitates prediction. This transformation is achieved by applying kernel trick with the widespread radial basis function (*RBF*) [39]. Section 4.3 discusses SVM training in greater detail.

After the training module has extracted all information regarding prediction from the samples, the information is applied in the online prediction module. This module quantizes the input trajectory at the beginning. The module then predicts possible locations and their corresponding probabilities using the quantized input trajectory, predictors' orders, and SVMs. The online prediction process discards the information unimportant for prediction to speed up the process, and decreases the bandwidth usage further. The process then combines the predicted locations into the final predicted locations. Each of these final predicted locations refers to one of the input time instants for prediction. Section 4.4 depicts the online prediction module.

4 Method Details

4.1 Vector Quantization

The first process in the training module is to construct a quantization mapping. The input to this mapping is an trajectory in its initial form, $Traj^*$, which is mentioned in Section 3, and in the form of $\{l^*_{n*}, l^*_{n*-1}, \ldots, l^*_1\}$. We denote this mapping as $Q_L(\cdot \mid N_{cell})$.$Q_L(\cdot \mid N_{cell})$ converts every position of $Traj^*$ into one of the predefined discrete cells or locations where N_{cell} is the number of the discrete locations. N_{cell} is a predefined parameter, and fixed throughout the quantization. This parameter influences the square of location quantization error, L_{err}. $Q_L(\cdot \mid N_{cell})$ should minimize the expectation of L_{err}, which is denoted as $E_{Lerr}(N_{cell})$, to retain as much information as possible. Suppose there are N_s sample trajectories which are $Traj^*_1, Traj^*_2, \ldots,$ and $Traj^*_{Ns}$, and the ith sample trajectory is notated as $Traj^*_i$, $E_{Lerr}(N_{cell})$ is evaluated empirically using:

$$E_{Lerr}(N_{cell}) = \sum_{i=1}^{N_s} \sum_{j=1}^{n_i^*} \left\| l^*_{i,j} - Q_L\left(l^*_{i,j} \mid N_{cell}\right) \right\|^2 \Bigg/ \left(\sum_{i=1}^{N_s} n_i^* \right), \quad (1)$$

where n_i^* is the number of positions in $Traj^*_i$, and $l^*_{i,j}$ is the jth position in $Traj^*_i$. Minimizing $E_{Lerr}(N_{cell})$ becomes a classical problem of VQ in lossy data compression. The positions in every $Traj^*$ form the training sample set, and N_{cell} refers to the *bit per vector* in the VQ problem, which is equal to $log_2(N_{cell})$. The proposed predictor employs *LBG* VQ algorithm [40] to solve the problem due to its simplicity and popularity, although applying other VQ algorithms is possible.

The proposed prediction method replaces all positions in every sample trajectory, $Traj^*$, with the discrete locations that are specified by $Q_L(\cdot \mid N_{cell})$. $Traj^*$ is then shrunk by combining consecutive identical locations into one discrete location. Each of the discrete location pertains to a duration value which is the residence time at the location. This value is the product of the number of positions combined and the sampling period for $Traj^*$. $Traj^*$ becomes a sequence of n tuples, $\{(l_n, \boldsymbol{d}^*_n), (l_{n-1}, \boldsymbol{d}^*_{n-1}), \ldots, (l_1, \boldsymbol{d}^*_1)\}$, where $n \leq n^*, l_i$ is the ith discrete location, and $\boldsymbol{d}^*_i$ is the duration value for l_i. Similar to $Q_L(\cdot \mid N_{cell})$, a quantization mapping, $Q_D(\cdot \mid N_{dur})$, is used to convert every duration value in the shrunk $Traj^*$ into one of the predefined discrete duration values, where N_{dur} is the number of the discrete duration values. Again, the proposed method minimizes the expectation of the square of duration quantization error, which is notated as D_{err}. Analogous to $E_{Lerr}(N_{cell})$, this expectation, $E_{Derr}(N_{dur})$, is found by

$$E_{Derr}(N_{dur}) = \sum_{i=1}^{N_s} \sum_{j=1}^{n_i} \left\| \boldsymbol{d}^*_{i,j} - Q_D\left(\boldsymbol{d}^*_{i,j} \mid N_{dur}\right) \right\|^2 \Bigg/ \left(\sum_{i=1}^{N_s} n_i \right), \quad (2)$$

where n_i is the number of locations in shrunk $Traj^*_i$ and $\boldsymbol{d}^*_{i,j}$ is the jth duration value in $Traj^*_i$. LBG is again used to construct $Q_D(\cdot \mid N_{dur})$. Using $Q_L(\cdot \mid N_{cell})$ and $Q_D(\cdot \mid N_{dur})$, every $Traj^*$ is transformed into $Traj$, which is a sequence consisting of n states, and expressed as $\{S_n, S_{n-1}, \ldots, S_1\}$. The choices of N_{cell} and N_{dur} depend on the available resources for prediction. More resources allow larger N_{cell} and N_{dur}, which lead to lower quantization errors. The

ith state, S_i, in the sequence is a tuple composed of a discrete location, l_i, and a discrete duration value, d_i. S_n represents the latest state, and S_1 represents the oldest one. The trajectory prediction problem becomes the estimation of the next state, S_{n+1}.

Some notations in this section are used again in the remainder of this paper. Readers may find a summary of notations useful. We place the summary in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2012.214.

4.2 Order Trimming

Some studies, such as in [30], employ conditional entropy to consider the appropriate order of a predictor, which is notated as k, but these studies are not suited to trajectory prediction utilizing precise spatiotemporal samples. The conditional entropy treats every state in $Traj$ as a state completely distinct from others. This treatment is adequate for lossless compression but not for trajectory prediction. It is because the extent of distinction or similarity between two states varies with the difference between their positions and residence durations. Employing conditional entropy does not take the similarity into account, and makes k higher than necessary. Therefore, the proposed method applies another approach that relates k with prediction accuracy. This approach makes k small while achieving low prediction error. Low prediction error is accomplished by reducing the expectation of the error of an optimal estimation. The optimal estimation is the mean of the random variable being predicted, which is the conditional expectation of l_{n+1} and d_{n+1}, given the sequence of last k states, $\{S_n, S_{n-1}, \ldots, S_{n-k+1}\}$. The condition expectation is $E[E[||S - E[S \mid S_n, \ldots, S_{n-k+1}]||^2]]$, where S is a vector, $(s_1, s_2, s_3)^T$, representing l_{n+1} and d_{n+1} simultaneously, i.e., $s_1 = x$ coordinate of l_{n+1}, $s_2 = y$ coordinate of l_{n+1}, and $s_3 = d_{n+1}$. Thus, the appropriate k is the smallest integer such that $E[E[||S - E[S \mid S_n, \ldots, S_{n-k+1}]||^2]]$ is lower than a desirable value, which is notated as E_{max}.

The proposed method uses a greedy approach to compute k. This method first sets k to be 1, and evaluates $E[E[||S - E[S \mid S_n, \ldots, S_{n-k+1}]||^2]]$. The method then checks if this expectation is lower than E_{max}. If not, the value of k is incremented by 1. The proposed method evaluates and checks the expectation again. This process iterates till a value of k is discovered such that the expectation is lower than E_{max}. This greedy approach is valid only if prediction accuracy increases with k generally. It can be verified by proving the following:

$$E\big[E\big[(s_i - E[s_i \mid S_n, \ldots, S_{n-p+1}])^2\big]\big] \geq E\big[E\big[(s_i - E[s_i \mid S_n, \ldots, S_{n-q+1}])^2\big]\big], \quad (3)$$

for $i = 1, 2, 3$, where $p, q \in \mathbb{N}_1$ and $p < q$. The proof is presented in Appendix B, which is available in the online supplemental material.

To evaluate $E[E[||S - E[S \mid S_n, \ldots, S_{n-k+1}]||^2]]$, it requires the probabilities, $Pr(S \mid S_n, \ldots, S_{n-k+1})$ and $Pr(S_n, \ldots, S_{n-k+1})$, for all the possible values of S and sequences $\{S_n, \ldots, S_{n-k+1}\}$. However, these probabilities are not explicitly available because we do not assume any fixed model. Instead, the proposed method applies the empirical probabilities based on the available samples. As Section 3 mentions, k depends on the current state of MS, and therefore we find k for each possible value of S_n. A value, k_{max}, is defined, which is the maximum value of k allowed, and is used to bound computation.

4.3 SVM Training

This section describes how to apply training samples to SVM. Every trajectory sample is initially in form of a sequence of regularly sampled position as described in Sections 3 and 4.1, and transformed into a trimmed $Traj$ with k states or tuples using VQ and order trimming, which are introduced in Sections 4.1 and 4.2. Each sample for SVM training is a trimmed $Traj$ plus its next state, which is a sequence of $k+1$ states. The training samples then become $\{S_{n+1}, S_n, \ldots, S_{n-k+1}\}$ or $\{(l_{n+1}, d_{n+1}), (l_n, d_n), \ldots, (l_{n-k+1}, d_{n-k+1})\}$. We denote a sample trajectory for SVM training as Seq in this section. Readers can find some basic concepts about SVM in Appendix C1, which is available in the online supplemental material. SVM's extension to multiclass classification is introduced in Appendix C2, which is available in the online supplemental material, which applies the results in [37] and [38].

It is possible to construct one single SVM to estimate $Pr(S_{n+1} \mid X)$, where X is the sequences of the last k states, $\{S_n, \ldots, S_{n-k+1}\}$. However, the complexity of the online prediction would be very high because the complexity increases with the number of pairwise class membership probabilities. The pairwise class membership probabilities are introduced in Appendix C2, which is available in the online supplemental material, and they are defined as $Pr(S_{n+1} = i \mid S_{n+1} = i \text{ or } j, X)$ and $Pr(S_{n+1} = j \mid S_{n+1} = i \text{ or } j, X)$, where i and j are any distinct and possible values of S_{n+1}. The number of possible values of S_{n+1} (or the number of classes in SVM terminology), N_c, is large in one single SVM due to a lot of possible values for S_{n+1}. A large value of N_c then leads to a large number of pairwise class membership probabilities and a high complexity. The high complexity is also applied to SVM training. Therefore, the proposed method splits the training samples, and constructs multiple SVMs from smaller training sets. Each SVM has a smaller N_c, and speeds up the online prediction and training.

The proposed method groups the samples for SVM training. The grouping turns a large set of Seq into many small subsets so that N_c in each subset is small. This method groups the Seq with the same current location, l_n. Each subset of samples is applied to training an SVM and each of the SVMs refers to a particular l_n. It is efficient because it only examines one component, S_n, in Seq for grouping. Besides, there is a high coherence between the next state, S_{n+1}, and l_n. The number of possible value of S_{n+1} is reduced, and hence smaller N_c. It is better to include more samples for training each SVM to increase prediction reliability. Therefore, for each SVM, other Seq with similar l_n to the SVM's l_n are also included. The differences among all samples' l_n in training the same SVM are defined to be not larger than R meters. R then becomes a tunable parameter in our prediction method. However, the lengths of each Seq in a sample subset may be different because

their values of k, which are determined by order trimming, are different. These values are adjusted so that all Seq in the same sample subset have the same k, and SVM training is then possible. We choose the maximum k among all Seq's k in the same subset as the new k for all Seq in the subset. Every Seq is adjusted according to its new k. After training every SVM, the proposed method indexes the SVMs using their l_n.

4.4 Online Prediction

The online prediction starts with quantizing and trimming the input trajectories to the prediction module as discussed in Sections 3, 4.1, and 4.2. A quantized and trimmed input trajectory is denoted as $Traj_{test}$, which is initially obtained by quantizing and trimming the input trajectory in Fig. 1. $Traj_{test}$ becomes longer, and the number of $Traj_{test}$ increases during the prediction process, which will be discussed shortly in this section. Another input to the online prediction is the time instants which are the times at when predicted locations are, and denoted as T. T is defined as the time which has elapsed since the start of $Traj_{test}$. The online prediction process is performed step by step as follows:

1. A $Traj_{test}$ is obtained by quantizing and trimming the input trajectory to the predict module.
2. The online prediction applies SVM to find out all the possible values of $Traj_{test}$'s next state and their probabilities.
3. The predictor then discards some of the possible values which have a low chance of being $Traj_{test}$'s next state to reduce computation.
4. The prediction process continues by extracting the possible values of $Traj_{test}$'s next states which will occur at the time T. The location values in the extracted states and their corresponding probability values are cached.
5. The predictor checks whether the prediction process should finish. If all the possible values of $Traj_{test}$'s next state are discarded or extracted, go to step 7. If there is still any possible value of $Traj_{test}$'s next state remaining, go to step 6.
6. The predictor attaches the remaining states to $Traj_{test}$ to form a new set of $Traj_{test}$, and goes back to step 2. $Traj_{test}$ becomes longer, and the number of $Traj_{test}$ increases.
7. The cached position and probability values are used in computing the final predicted location at T. The predicted location is output as the final prediction result only if the result is found to be reliable.

Readers may notice that performing the process from steps 2 to 6 is iterative. We explain more about the steps 2 and 3 in Section 4.4.1. Section 4.4.2 discusses the steps 4, 5, and 6. The last step is described in Section 4.4.3.

4.4.1 Predict Next States of Trajectories

The proposed predictor evaluates every possible value of $Traj_{test}$'s next state and their corresponding conditional probability, given the input trajectory to the prediction module. We represent $Traj_{test}$ as a sequence of states, $\{S_{n+m}, S_{n+m-1}, \ldots, S_{n-k+1}\}$, and $m \in \mathbb{N}_0$, which is the total number of predicted states on $Traj_{test}$. Initially, there is no predicted states on $Traj_{test}$, and $m = 0$. The conditional probability of $Traj_{test}$'s next states, S_{n+m+1}, given the initial $Traj_{test}$, is found by

$$\begin{aligned} &Pr(\mathrm{S}_{n+m+1} \mid \boldsymbol{S}_n, \boldsymbol{S}_{n-1}, \ldots, \boldsymbol{S}_{n-k+1}) \\ &\quad = \sum_{\forall\{\boldsymbol{S}_{n+m}, \boldsymbol{S}_{n+m-1}, \ldots, \boldsymbol{S}_{n+1}\}} Pr(\boldsymbol{S}_{n+m+1}, \boldsymbol{S}_{n+m}, \ldots, \boldsymbol{S}_{n+1} \mid \boldsymbol{S}_n, \\ &\qquad \boldsymbol{S}_{n-1}, \ldots, \boldsymbol{S}_{n-k+1}) \end{aligned} \tag{4}$$

where the initial $Traj_{test}$ is the sequence, $\{S_n, S_{n-1}, \ldots, S_{n-k+1}\}$, and $\{S_{n+m+1}, S_{n+m}, \ldots, S_{n+1}\}$ is a possible future trajectory. The term, $Pr(S_{n+m+1}, S_{n+m}, \ldots, S_{n+1} \mid S_n, S_{n-1}, \ldots, S_{n-k+1})$, on the R.H.S. of (4) is evaluated by

$$\begin{aligned} &Pr(\boldsymbol{S}_{n+m+1}, \boldsymbol{S}_{n+m}, \ldots, \boldsymbol{S}_{n+1} \mid \boldsymbol{S}_n, \boldsymbol{S}_{n-1}, \ldots, \boldsymbol{S}_{n-k+1}) \\ &\quad = \prod_{i=1}^{m+1} Pr(\boldsymbol{S}_{n+i} \mid \boldsymbol{S}_{n+i-1}, \ldots, \boldsymbol{S}_{n-k+1}), \end{aligned} \tag{5}$$

where $Pr(S_{n+i} \mid S_{n+i-1}, \ldots, S_{n-k+1})$ on the R.H.S. of (5) is found by the SVM introduced in Section 4.3.

S_{n+m+1} may have numerous possible values. The computation for probability evaluation, which uses (4) and (5), and finalizing prediction results, which is discussed in Section 4.4.3, can be huge. To avoid numerous states, the proposed method discards state values which are insignificant for prediction. A discarded state value, $S_{n+i} = s$, should have the conditional probability, $Pr(S_{n+i} = s \mid S_{n+i-1}, \ldots, S_{n-k+1})$, which is one of the terms in (5), lower than a very small value. This value is predefined and denoted as Pr_{min}.

4.4.2 Extract Possible Future Locations

The prediction process examines the duration from the start of $Traj_{test}$ to one of its states, S_{n+m+1}, which are introduced in Section 4.4.1. The duration, D_{n+m+1}, is computed by

$$D_{n+m+1} = d_{n+m+1} + d_{n+m} + \cdots + d_{n-k+1}. \tag{6}$$

If a value of S_{n+m+1} satisfies the condition $D_{n+m+1} \geq T$, we will cache the value of l_{n+m+1} in S_{n+m+1} and its corresponding probability, which is evaluated by (5) and (6). All cached data are used in finalizing prediction results, which is discussed in Section 4.4.3. The cached data are actually the possible future locations at the time T and their probabilities.

For every value of S_{n+m+1} which satisfies $D_{n+m+1} < T$, the predictor put that value at the front of $Traj_{test}$ to form a new $Traj_{test}$, which is $\{S_{n+m+1}, S_{n+m}, S_{n+m-1}, \ldots, S_{n-k+1}\}$. A set of new $Traj_{test}$ is then computed. The online prediction then goes back to the step 2 of the process, and applies the new $Traj_{test}$. m is incremented by 1 when the step 2 is performed again. If there is no value of S_{n+m+1} which satisfies $D_{n+m+1} < T$, the process finalize the prediction based on the cached data.

4.4.3 Finalize Prediction Results

The online prediction process finalizes prediction results using all the cached possible future locations at the time T and their probabilities. The optimal prediction is the conditional expectation, $E[loc(T) \mid S_n, S_{n-1}, \ldots, S_{n-k+1}]$, where $loc(T)$ is the future location at T to be computed. The

evaluation of the expectation is simply the weighted sum of the cached location based on their corresponding probabilities. Since the prediction process discards some possible locations in step 3, those probabilities need to be normalized.

However, the expectation may not be a reliable predictor if the trace of the covariance of $loc(T)$, which is expressed as $Var(T)$, is large. The proposed predictor rejects unreliable prediction results, and does not send out those results to networks. Thus, more bandwidth is saved. More specifically, if $Var(T)$ is larger than a predefined value, Var_{max}, the prediction result should not be output. Instead, the proposed predictor estimates the future location at the time T to be as same as the last predicted location. The percentage of rejecting prediction results increases with T and the randomness of the MS movements typically. In our experiments, it ranges from 5 to 75 percent approximately.

5 Results and Discussions

5.1 Experimental Setup

This study simulates 100 MSs travelling in an environment, which dimension is 3,500 m $\times$ 3,500 m. Each MS is associated with a movement pattern. Each movement pattern is a trajectory generated by a mobility model. The format of the generated trajectories is a sequence of regularly sampled locations as described in Section 3. Section 5.2 explains how a movement pattern is generated. The experiments in our study resemble to collecting 100 MSs' past trajectories within a period of time for training predictors and applying the trained predictors online. Each MS may travel in the environment many times, and generate many trajectories. The differences among the trajectories from the same MS are emulated by adding Gaussian noise to the sample positions in the MS's movement pattern, which is a trajectory generated by a mobility model. A large noise value makes a trajectory differ with others a lot. The noise is sometimes referred as the uncertainty or randomness of a trajectory. Experiments under high, medium, and low uncertainty are conducted. These levels of uncertainty correspond to noise variances of 2,000, 1,000, and 500, respectively. The uncertainty is composed of the position measurement noise and arbitrariness of MS routes (i.e., an MS may take a route different from its regular routes occasionally). We assume these two components are independent, and follow Gaussian distribution. A total of about 60,000 trajectories are generated for our experiments. The average duration of a trajectory is about 8,000 s.

We compare the proposed predictor with other predictors in this section. For a clear and concise comparison, we do not include every prediction method or approach. Our experiments only cover methods or approaches which are as follows:

1. The methods are easily adoptable in predicting MS precise trajectories generally, where the trajectories are not expressed by large cells or regions (on the order of km^2), and not restricted to specific movement patterns.
2. They are able to estimate the future MS locations that will be visited many minutes later. Methods that can only predict milliseconds ahead are excluded because their application in mobile computing is limited.
3. They are popular in mobile computing or other disciplines. The popularity can be revealed by their citation rate, or equivalently, how commonly they are found in other studies.
4. They have been shown to be one of the best among the methods that share similar approaches. Thus, other methods which apply similar approaches are omitted for concise comparisons.

Predictors that apply Markov chain modeling with finite order are found in many studies in mobile computing [1], [4], [5], [27], [28], [29]. Among these predictors, we choose the second-order Markov model as the first predictor in our comparison because we rarely find a Markov chain predictor with an order greater than 2. The second predictor in our comparison is the one in Bhattacharya's work [30], which is an enhanced LZ78 method. That work has gained a high citation rate, and is one of the best among the predictors employing compression algorithm [26]. The third predictor is the one applies trajectory similarity matching. This approach can be easily found in mobile computing and other areas [32], [33], [35], [36]. It can be shown that this approach achieves high prediction accuracy as compared to other machine learning approaches [34]. Among all the predictors from similarity matching approach, we apply the method in [35] and [36], which is easily adoptable in predicting MS precise trajectories. The predictors employing deterministic models, which are introduced in Section 2.1, are excluded. Those methods are only appropriate for predicting milliseconds ahead and specific movement patterns.

The first two predictors and our proposed method need quantize the positions on trajectories and the residence durations at these positions, as Sections 2.2 and 4.1 mention. We name these *quantization-based* predictors. The number of the discrete locations and duration values for the quantization are notated as in Section 4.1, which are N_{cell} and N_{dur}, respectively. For the third predictor, the input trajectories for this approach are in form of sequences of positions, which are sampled regularly and not quantized. This approach groups the sample trajectories into clusters using K-means in training. The number of clusters and sampling period is denoted as $N_{cluster}$ and T_{sample}, respectively. For brevity, we label the first three predictors as "*Second-Order*," "*LZ78*," and "*Clustering*," and our proposed method as "*Proposed*." All experiments are run on a 64-bit machine with an Intel Core i7-740QM CPU and 8-GB RAM. Appendix D, which is available in the online supplemental material, describes the first three predictors in more detail.

5.2 Mobility Models

This study generates the movement patterns randomly by three mobility models in [41], [42], and [43] for the experiments, and we name these models as *Kim*, *Mei*, and *Rhee* in this paper, respectively. These models are based on real trajectory traces, and, therefore, they are adopted by

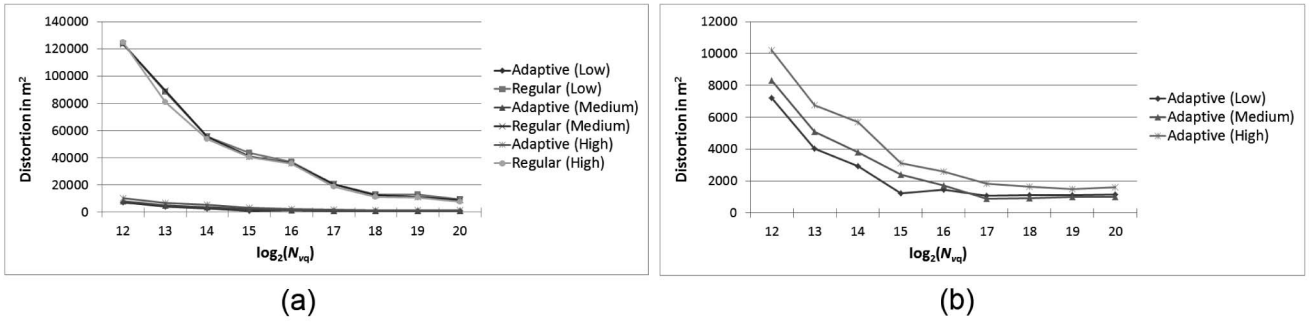

Fig. 2. Quantization performance of (a) both approaches and (b) adaptive approach.

this study. Each model contributes one-third of sample trajectories to every experiment. These models base on a phenomenon that an MS tends to be stationary at some destinations for some time, followed by a long distance of travel or flight to other destinations, and becomes stationary again. We distribute 100 destinations randomly in the environment of the experiments. We assume these destinations are circular regions. Their radii are also determined by chance, which are between 5 and 20 m. Every trajectory randomly starts at one of the destinations. Appendix E, which is available in the online supplemental material, shows the formulations of these three models and their parameter values. These values match with those parameter values in [41], [42], and [43], which derives from real-world samples.

5.3 Quantization Distortion

In this section, we examine the quantization distortion due to quantizing positions and residence duration. This quantization is essential to quantization-based predictors. A common and intuitive approach is to quantize the positions based on evenly distributed cell like the quantization in [28], and quantize the durations regularly. We name this *regular approach*. It contrasts to the proposed approach, as Section 4.1 depicts, which quantizes the positions and durations adaptively according to given statistics. We call this *adaptive approach*. These quantized positions and durations are used in forming states or tuples that comprise a quantized trajectory. The experiments in this section compare both approaches' quantization distortion. The better approach should attain a lower quantization distortion under similar N_{cell} and N_{dur}. The distortion is defined as the average mean square of quantization error of quantizing every sample locations taken from all the trajectories. These locations are sampled at a rate of one sample per 100 s. We show the results under different levels of trajectory uncertainty in Fig. 2. The data points in this figure are obtained by varying N_{cell} and N_{dur} from 2^6 to 2^{10}. The distortions of both approaches are plotted against $log_2(N_{vq})$, where N_{vq} is the product of N_{cell} and N_{dur}. $log_2(N_{vq})$ indicates the number of bits for representing a discrete state in a quantized trajectory.

Fig. 2a reveals that finer quantization decreases the distortion in general. However, finer quantization cannot reduce the distortion much when N_{vq} becomes large. A very fine quantization has already involved enough number of discrete values for low distortion, and therefore introducing more discrete values to the quantization mapping cannot enhance the quantization significantly. Also, the adaptive approach outperforms the regular approach under every level of uncertainty. It is because the adaptive approach aims to minimize the expectation of the square of quantization error, as Section 4.1 explains, but the regular approach does not.

The influence of the uncertainty level on the regular approach is not as much as on the adaptive approach. Fig. 2b examines the adaptive approach closer, and we can see that the distortion of this approach increases with the uncertainty level. The adaptive approach is sensitive to the uncertainty level because the uncertainty affects the distribution of MS positions and residence durations, on which this approach depends. Besides, higher uncertainty means the samples for quantization disperse more, and the adaptive approach requires a larger N_{vq} to attain low distortion.

To provide a fair comparison among the quantization-based predictors, all of these predictors in every following experiment in this paper adopt the adaptive approach.

5.4 Accuracy

This section compares the proposed predictor's accuracy with other predictors'. Predictions with various *prediction lengths*, which are the time differences between the current and predicted locations, are performed. Prediction lengths determine the input time instants, T, to the online prediction, which is introduced in Section 4.4. We tried the prediction lengths at every 100 s between 100 s and 5,000 s, i.e., 100 s, 200 s, 300 s, ..., 4,900 s, 5,000 s, in our experiments. The gap between two consecutive prediction lengths is called *prediction step*, which is 100 s in this study. We also apply various values to the parameters of quantization-based predictors and clustering predictor. These values are stated in Table 1.

TABLE 1
Parameter Values of Predictors

N_{cell}	2^9, 2^8, 2^7, 2^6
N_{dur}	2^8, 2^7, 2^6
$N_{cluster}$	5k, 2k, 500
T_{sample}	1000s, 500s, 100s, 10s

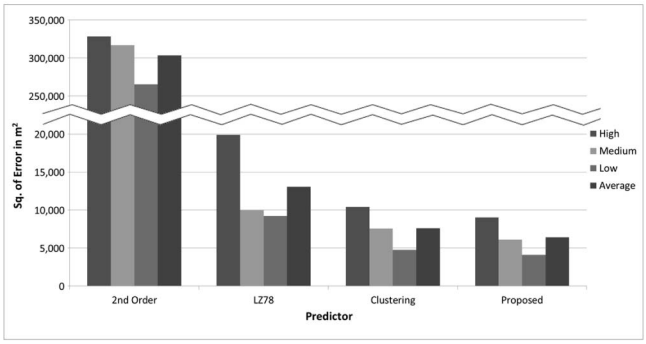

Fig. 3. Prediction error of each predictor.

Fig. 3 presents the comparison between the accuracy of the proposed predictor and the others. The accuracy is in terms of the average square of prediction error in every prediction performed. The errors at different uncertainty levels are also investigated. To achieve a fair comparison, we select the parameter values from Table 1 that lead to maximum accuracy for each predictor. Those values are listed in Table 2. The parameter values specific to the proposed predictor are shown in Table 3.

Fig. 3 shows that the second-order predictor suffers from low prediction accuracy. However, LZ78 has significantly lower errors. This observation implies that higher order in Markov chain modeling reduces prediction errors. It is contrary to the common argument that second order is sufficient for prediction and any further order does not reduce the errors much [1], [28], [31]. As Section 2 mentions, Markov chain of very few orders cannot capture all characteristics of user movements for prediction, especially for small cells which have radii of few meters. On the other hand, if the cell is huge, for example, a typical GSM cell, a low order of Markov chain predictor suffices.

We can also observe that LZ78 performs worse than the other two predictors and is not robust to high uncertainty. It is because the quantization is fine in some areas. Many possible sequences pertaining to those areas, which may appear in online prediction, are not discovered from the training samples in LZ78. This problem becomes serious when the uncertainty level is high. Another observation is that the proposed predictor is more accurate than the clustering predictor when we observe this figure carefully. The future MS movement may correlate more to some segments of a sample trajectory. The most recent part of a past trajectory usually has a stronger coherence with the next location to be visited than the other parts. The clustering method does not consider this coherence, but this is considered by the proposed method. The feature

TABLE 2
Parameter Values for Maximum Accuracy

2nd Order	$N_{cell} = 2^9$; $N_{dur} = 2^6$
LZ78	$N_{cell} = 2^8$; $N_{dur} = 2^7$
Clustering	$N_{cluster}$ = 2k; T_{sample} = 10s
Proposed	$N_{cell} = 2^8$; $N_{dur} = 2^7$

TABLE 3
Parameters of Proposed Predictor

E_{max}	5
k_{max}	25
R	200m
Pr_{min}	0.5%
Var_{max}	1000000

space of the proposed predictor reflects the relationship between the past and future locations better than euclidean distance does. The proposed method is even more superior to the clustering method when bandwidth usage is considered, as the next section discusses.

5.5 Accuracy versus Bandwidth Usage

This section investigates the bandwidth usage for prediction in byte per second (*Bps*). The bandwidth usage of predictors is categorized into two types: outgoing and incoming (from the view of the predictor), which have been introduced in Section 3. They are discussed in Sections 5.5.1 and 5.5.2, respectively. We plot the minimum required bandwidth against the average square of prediction error and show the results in Fig. 4 (for outgoing bandwidth) and

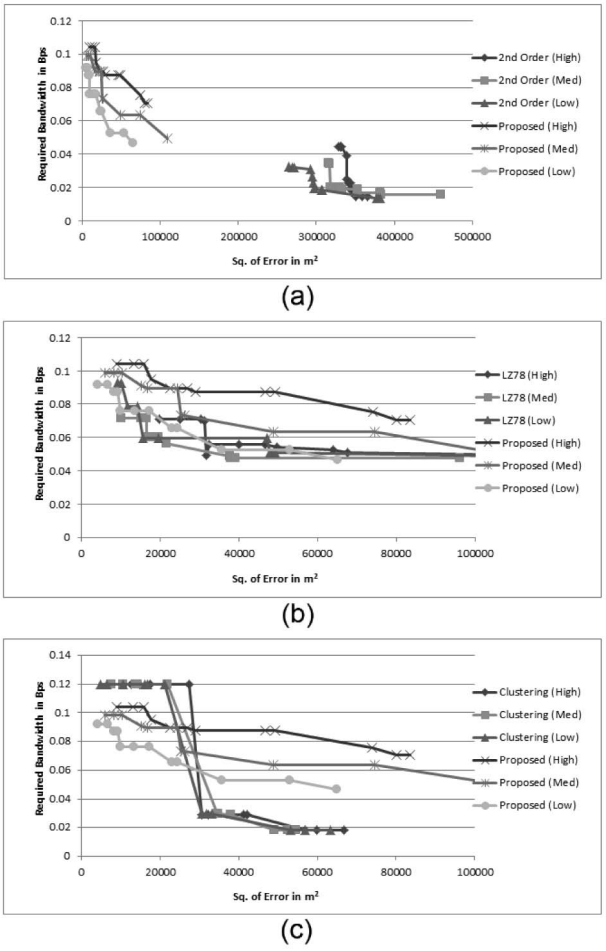

Fig. 4. Compare the outgoing bandwidth of the proposed predictor with the (a) second order (b) LZ78, and (c) clustering predictor.

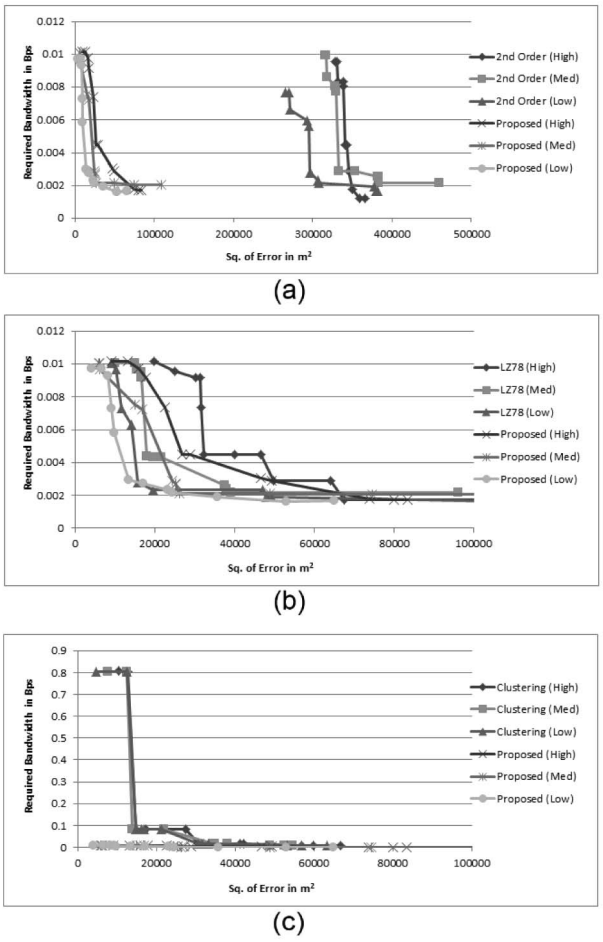

Fig. 5. Compare the incoming bandwidth of the proposed predictor with the (a) second order (b) LZ78, and (c) clustering predictor.

Fig. 5 (for incoming bandwidth). Each of these figures compares the proposed predictor with one of the other three predictors, i.e., second order, LZ78, and clustering, at different uncertainty levels. The data points of the graphs in these figures are obtained by applying every parameter value in Table 1. Any protocol overhead is ignored in our experiments so that the results are independent of adopting any protocol. If the graph of a predictor is close to the y-axis, it implies high accuracy. If the graph is near to the x-axis, it means low bandwidth usage.

5.5.1 Outgoing Bandwidth

In our experiments, a prediction result is represented by 12 bytes: 8 bytes for the predicted position and 4 bytes for the time instant for this position. The outgoing bandwidth usage increases with the number of prediction results sent out from the predictors. To save more bandwidth, the predictors at MSs do not send out any prediction result to the server if the current prediction result is as same as the previous one. At the server side, the previously received prediction result is used in the current prediction if the current prediction result is not available.

The average outgoing bandwidth usage is the quotient of dividing the average amount of data sent from the predictor at a MS by the maximum prediction length, which is 5,000 s in the experiments. Those data represent the prediction results. As we can see Fig. 4, every predictor in the experiments requires more bandwidth for a higher uncertainty level while maintaining the same accuracy level. It is because the difficulty in prediction increases with the uncertainty level. More information, and thus bandwidth are required for high accuracy.

We also find that the second-order predictor makes huge prediction errors in Fig. 4a, which is consistent with the results in Section 5.4. However, its bandwidth usage is lower than the proposed predictor's. This result implies the second-order predictor is conservative, and tends to estimate MSs to be stationary, which suspends the transmission of the prediction results from the predictor. It is due to the lack of input information for prediction in low-order Markov chain model.

More input information is available in LZ78. Therefore, LZ78 is a more aggressive predictor, and more outgoing bandwidth is consumed, as Fig. 4b shows. However, its bandwidth usage is still slightly lower than the proposed predictor's. LZ78 degrades to be a low-order Markov chain predictor sometimes, which is similar to the second-order method. It is because not every possible sequence can be discovered in training, as Section 5.4 mentions. Therefore, the graph of the proposed predictor is closer to the y-axis generally, and the LZ78 predictor is not as accurate as the proposed predictor is.

In Fig. 4c, it is found that the clustering predictor achieves the accuracy that the proposed predictor does. It also demonstrates a lower bandwidth usage than the proposed predictor does at high error level. However, for low prediction errors, its required bandwidth surges to a high level. The absence of quantization in the clustering predictor causes great bandwidth usage when T_{sample} is small. Anyhow, the bandwidth usage is capped at 0.12 Bps. It is because the maximum output bandwidth must be the data size for a prediction result per prediction step in our experiments, which is 12 bytes per 100 s.

5.5.2 Incoming Bandwidth

The average incoming bandwidth usage shown in Fig. 5 is composed of two parts. The first part is providing input to the online prediction. This part is simply the quotient of dividing total amount of MS trajectory data sent to the predictor at server by the sum of all trajectories' durations. The second part is sending data to and from the predictor at server for training purpose. The training data are MS trajectory samples, which is sent from MS clients to the server. For the quantization-based predictors, we also need to send quantization maps, which are produced by the training, from server to every client. The number of MS trajectory samples for training is 10,000 in our study. The trajectory samples for training are either sequences of quantized states in quantization-based predictor, or locations with time stamps in clustering predictor.

We assume that the period of updating the quantization map and retraining predictors is 1 month in our experiment. It is because the road topologies in the real world are unlikely to change enormously every day or week. 1 month is often long enough for an area of a few km^2 to accumulate 10,000 sample trajectories (about 333 per day). Therefore, the total number of MS trajectory for training predictors is

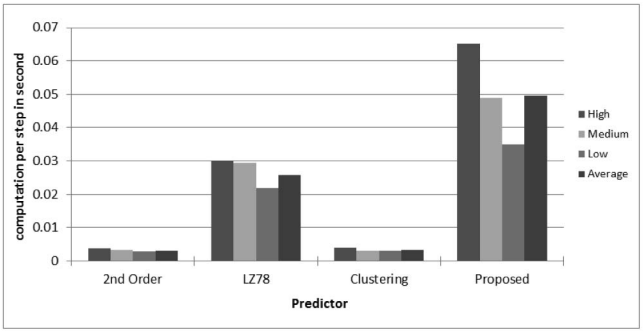

Fig. 6. Computation of performing prediction for each predictor.

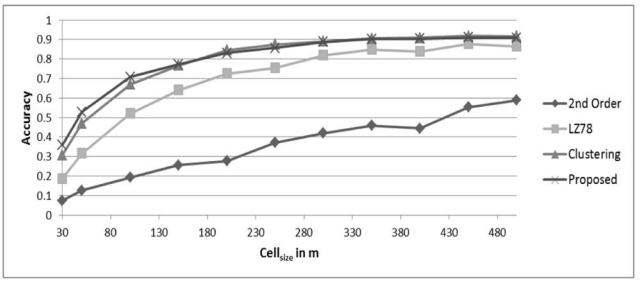

Fig. 7. Average paging accuracy of each predictor.

set to be 10,000. Appendix F, which is available in the online supplemental material, gives more information about the bandwidth usage for training predictors. The appendix, which is available in the online supplemental material, shows that the bandwidth usage for training predictors is generally insignificant.

First, we compare the second-order predictor with the proposed predictor. Fig. 5a indicates the prediction error of the second-order predictor is much higher regardless of the incoming bandwidth usage. This huge difference in accuracy has been already explained in Section 5.4. Second, the LZ78 predictor is compared with the proposed predictor. Like the results in Section 5.5.1, Fig. 5b shows that the LZ78 predictor is not as accurate as the proposed predictor is. The graph of the proposed predictor is closer to the y-axis in general. The LZ78 predictor is also noticeably worse than the proposed one when the uncertainty level is high. It is because the LZ78 predictor is not robust to high uncertainty, which has been shown in Section 5.4. In fact, the comparison among these three quantization-based predictors' incoming bandwidth can be solely based on their accuracy. It is because they have the same amount of incoming data representing the MS trajectories.

Last, the clustering predictor is examined in Fig. 5c. Its required input bandwidth is significantly higher than the proposed predictor, especially for low prediction error, although these predictors' accuracies are similar. It is because the proposed predictor employs VQ to greatly reduce the amount of the trajectory data sent from MSs. The results in this section and Section 5.5.1 demonstrate the positive effect of the quantization on reducing both incoming and outgoing bandwidth usage. Some readers may also notice that there is a sudden jump in the bandwidth usage for the clustering method. The jump ends at a value of square of error about 20,000, which refers to $T_{sample} = 100$ s. This observation implies that it is much harder to improve prediction accuracy when T_{sample} is lower than a certain value. As learned from the sampling theorem, we capture most information of a trajectory when T_{sample} is low enough, which is as low as about 100 s in our case. Accuracy improvement thus becomes much harder.

5.6 Computation

This section investigates the computation for online prediction in each predictor. It is measured in average amount of time spent per prediction performed. The computation under different uncertainty level is also examined. The results can be found in Fig. 6. The second order and clustering predictor shows low computation in all circumstances. The second-order predictor need not retrieve much information for prediction and the clustering predictor only retrieves a single trajectory for all the prediction of various prediction lengths. These reasons justify the low computations. On the other hand, the LZ78 and proposed predictor handle a huge amount of data for prediction, and they take more time to extract prediction results from the data. Their computations are also sensitive to the uncertainty level. It indicates that more data are involved in these predictors when the uncertainty level rises. Particularly, the proposed method has the highest computation. It is caused by the time spent on the membership probabilities estimation, which is depicted in Section 4.3, and considering multiple possible future locations, which is detailed in Section 4.4. Nevertheless, its computation is well below 1 s, and it is sufficient for online prediction if the prediction step is larger than 1 s.

If the proposed prediction method is performed at a mobile device, we may need to consider its average power consumption at the device due to the computation. However, we observe that the average power is insignificant as compared to the power due to the mobile device's system and OS. More details can be found in Appendix G, which is available in the online supplemental material.

5.7 Possible Applications

5.7.1 Paging

Some applications need an accurate prediction of the MSs' visited region or cell instead of exact position. A typical example is *paging*. Paging is to identify the cell that an MS is visiting. The percentage of correct prediction about the cell to be visited in the future is called *paging accuracy*. It is obvious that paging accuracy depends on the size of cells. In this section, paging accuracy is plotted against the cells' radii. The results are shown in Fig. 7. Each data point on the graph is taken from the average of accuracy value at high, medium, and low uncertainty level. We assume that the cell sizes are identical, and the cells are located in the environment evenly. We compare the paging accuracy with the prediction errors of the predictors in Fig. 3 to determine the prediction errors consistent with the accuracies in real applications.

As we can see from Fig. 7, the paging accuracy agrees with the results in Section 5.4, in which the second-order predictor is significantly less accurate than the other three

TABLE 4
Suggested $Cell_{size}$ for Some Types of AOI

AOI Type	Approximate Area	Suitable $Cell_{size}$
Football Pitch e.g., International Standard	830m^2	30m
Department Store e.g., Harrods	10000m^2	100m
Stadium e.g. Old Trafford Stadium	30000m^2	200m
Campus e.g., UCL in Central London	160000m^2	400m
Theme Park e.g., Disneyland in LA	350000m^2	500m

predictors enormously, and the proposed predictor and the clustering predictor are the most accurate. Although the accuracies of the proposed and clustering predictor are similar for large cells, we can clearly observe a difference for small cells (at least 10 percent difference). It is because both predictors achieve a low prediction error, as shown in Fig. 3, and their difference in accuracy can be only revealed in environments with small cells, whose radii are less than 100 m. However, when the cells are large, the predictors' paging accuracies become similar. This observation suggests that an environment with large cells does not demand a very accurate predictor. Therefore, if the cells are small, an accurate predictor is important. The required cell size depends on what kind of area of interest (*AOI*) the predictors target, where the radius of AOI is represented as $Cell_{size}$. For example, if we need the predictor estimate whether or when an MS visits a football pitch, $Cell_{size}$ should be as small as 30 m. On the contrary, $Cell_{size} = 500$ m suffices if the AOI is a theme park. Some types of AOI and their corresponding suitable cell sizes are shown in Table 4.

5.7.2 Information Feeding

In Section 5.7.1, we have introduced an application that is solely dependent on the predictors' accuracy. In this section, we proceed to discuss another application which is influenced by both the accuracy and bandwidth usage. Feeding mobile clients information according to their location is a popular application. Mobile operators can feed various types of the information, for example, traffic situation and advertisement. A conventional and passive approach is sending the information when the MS clients arrive at the designated AOI. Here, we show a proactive approach in which the information is sent ahead of the clients' arrivals at the AOI, which is similar to the early reminder application in [1]. This approach is useful if the clients are more preferable to make some decisions in advance, and the decisions are relevant to where they are going to visit or pass through. For instance, it is more desirable to send an advertisement of a product to a client before he has purchased another similar product. However, the price of this approach is a higher bandwidth usage. If the network predicts the future location of a client wrongly, it will send out inappropriate information, and need to resend something suitable afterward. Therefore, an accurate predictor with low bandwidth plays an extremely important role in this approach.

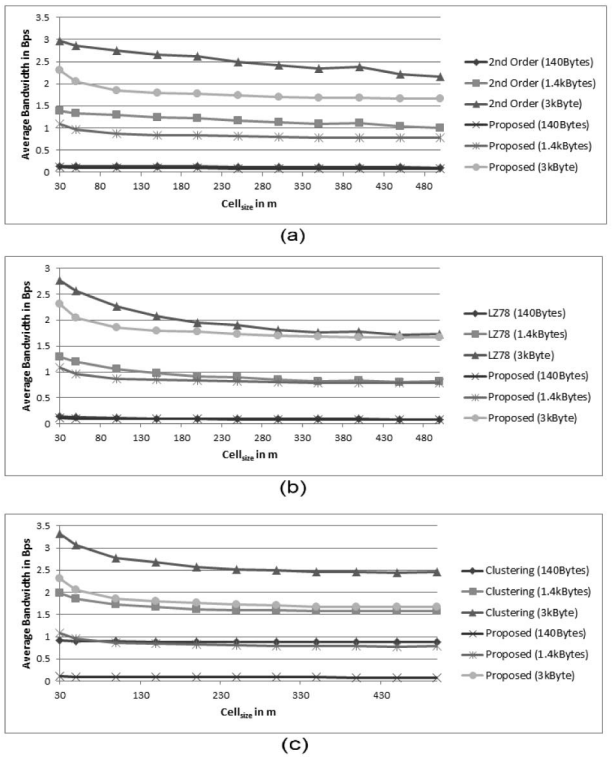

Fig. 8. Compare the average bandwidth usage in information feeding of the proposed predictor with the (a) second order (b) LZ78, and (c) clustering predictor.

To support the proactive information feeding, we set the prediction to be server-based. The experiment setup is basically as same as the one in Section 5.5.2. In addition, the environment consists of AOIs or cells of regular size and sharp, which is as same as the AOI setup in Section 5.7.1. Each AOI is related to a piece of message which is sent to any client visiting or predicted to visit that AOI. Each message only pertains to one AOI. The message size is denoted as Msg_{size}. The server predicts the cells to be visited by clients at T_{pred} seconds ahead, and sends the message pertaining to those predicted cells. After T_{pred} seconds, if the server finds that the prediction is incorrect, it will transmit the message to the clients according to their currently located cells. This procedure repeats periodically. The period of this procedure is denoted as T_{period}. The experiment results in this section can conclude whether the results in Sections 5.5.2 and 5.7.1 are applicable to this kind of applications.

After conducting this set of experiments, we plot the average bandwidth usage per MS for each predictor against $Cell_{size}$ at different Msg_{size}. The results are presented in Fig. 8. Several values of T_{pred}, T_{period}, and different levels of uncertainty (high, medium, low) are attempted in the experiments. The average results are shown in the graphs. The values of T_{pred} and T_{period} tried are: $T_{pred} = 1{,}000$ s, 2,000 s, 3,000 s, 4,000 s, 5,000 s, and $T_{period} = 15$ min, 30 min, 45 min, 60 min, 75 min, 90 min. The $Cell_{size}$ ranges from 30 to 500 m for $Msg_{size} = 140$ byte, 1.4 kB, 3 kB. 140 byte is about the size of one SMS, 1.4 kB can be a long SMS, and 3 kB can be an MMS with image and sound. The results are consistent with those in

Section 5.7.1. The overall trend for every predictor is that the bandwidth usage decreases when $Cell_{size}$ is larger due to a better accuracy as shown in Fig. 7. When $Cell_{size}$ is very large, this trend starts to stop because further improvement in accuracy is very limited.

The second-order predictor performs significantly worse than our proposed predictors because of its low accuracy. The clustering method also performs poor because it suffers from a high input bandwidth usage as explained in Section 5.5.2. Only the LZ78 predictor can perform comparably to the proposed method. However, as the message size increases, the proposed method outperforms the LZ78, and the difference becomes increasingly noticeable. This is because the importance of prediction accuracy is higher for a large message. There is a bandwidth penalty for any incorrect prediction, and this penalty is proportional to the message size. As we can learn from Section 5.7.1, the proposed predictor is a more accurate predictor. Another observation is that their difference in bandwidth usage increases when $Cell_{size}$ decreases. This again matches with the results in Fig. 7 where the accuracy difference between two predictors magnifies when $Cell_{size}$ is smaller.

5.7.3 Peer to Peer Networking

Another kind of possible applications employing location prediction is related to increasingly popular social networking applications in mobile version, for example, Facebook and Google+. The users may share their predicted locations with their friends or other users in the same peer group. The usefulness of this kind of applications is mentioned in [1]. For instance, a user schedules a meeting with all the people in the same peer group at a time and place which will be convenient to everyone.

The experiment setup is similar to the setup in Section 5.7.2. However, the information messages are sent from one MS client to other MS clients in the same peer group instead of from the server to each client. The total number of clients in a peer group is denoted as Gp_{size}. We set the Gp_{size} of every group to be as same as each other in our experiments. The results are presented in Fig. 9, which are graphs plotting average bandwidth usage against Gp_{size} at different Msg_{size}. In addition to T_{pred}, T_{period} and different levels of uncertainty, we try different values of $Cell_{size}$ which are 30, 50, and 100 m, and the average results are shown in the graphs.

The bandwidth usage increases linearly for every predictor in Fig. 9. This is because the number of message sent out by a client is directly proportional to the total member in the same peer group. Again, the second-order predictor performs the worst because of its poor accuracy as shown in Fig. 9a. In Fig. 9b, the LZ78 predictor performs better, although it is still worse than the proposed method. The difference in performance is less when the clustering method is compared with the proposed method as shown in Fig. 9c. These observations agree with the results in Fig. 7. If the predictors' accuracies are similar, the total messages sent out are similar too. However, we can still notice a trend that the difference becomes more noticeable when Gp_{size} and Msg_{size} increase. This trend suggests that the proposed predictor can be significantly better than other predictors when there are many messages and much data to send. In that case, even a little enhancement in prediction accuracy can make a great difference.

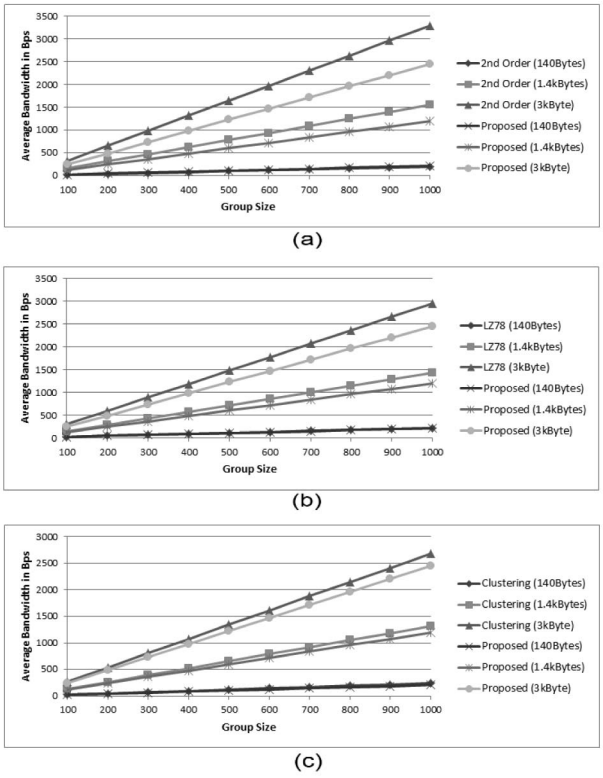

Fig. 9. Compare the average bandwidth usage in peer to peer networking of the proposed predictor with the (a) second order (b) LZ78 and (c) clustering predictor.

To sum up, all the results in Section 5.7 demonstrate that our proposed predictor is noticeably better than other predictors when the importance of prediction accuracy is significant. This phenomenon can be found in both client-based and server-based prediction. On the contrary, the LZ78 predictor's performance is obviously poorer than the proposed method when prediction accuracy is crucial. Also, the clustering method is inadequate for server-based prediction. The proposed method is, thus, an amenable and viable predictor, which can perform as the best generally.

6 Conclusion and Future Work

This paper presents a predictor that effectively utilizes massive spatiotemporal samples, and achieves high accuracy for bandwidth consumption lower than other predictors. We are now considering more factors in the prediction other than past trajectories, though this comes at the expense of more resources, such as networks, computation, and storage. This study can also be extended by exploring more applications which employ trajectory prediction.

References

[1] D. Ashbrook and T. Starner, "Using GPS to Learn Significant Locations and Predict Movement across Multiple Users," *Personal and Ubiquitous Computing*, vol. 7, no. 5, pp. 275-286, 2003.

[2] A. Harrington and V. Cahill, "Routing Profiling - Putting Context to Work," *Proc. ACM Symp. Applied Computing (SAC)*, pp. 1567-1573, 2004.

[3] S.Y. Cho and W.S. Choi, "Robust Positioning Technique in Low Cost DR/GPS for Land Navigation," *IEEE Trans. Instrumentation and Measurement*, vol. 55, no. 4, pp. 1132-1142, Aug. 2006.

[4] W.S. Soh and H. Kim, "A Predictive Bandwidth Reservation Scheme Using Mobile Positioning and Road Topology Information," *IEEE/ACM Trans. Networking,* vol. 14, no. 5, pp. 1078-1091, Oct. 2006.
[5] E. Cayirci and I. Akyildiz, "User Mobility Pattern Scheme for Location Update and Paging in Wireless Systems," *IEEE Trans. Mobile Computing,* vol. 1, no. 3, pp. 236-247, July-Sept. 2002.
[6] P. Pathirana, A. Savkin, and S. Jha, "Mobility Modelling and Trajectory Prediction for Cellular Networks with Mobile Base Stations," *Proc. ACM MobiHoc,* pp. 213-221, 2003.
[7] G. Box, G.M. Jenkins, and G. Reinsel, *Time Series Analysis: Forecasting and Control,* fourth ed. Wiley, 2008.
[8] Z. Zaidi and B. Mark, "Mobility Tracking Based on Autoregressive Models," *IEEE Trans. Mobile Computing,* vol. 10, no. 1, pp. 32-43, Jan. 2011.
[9] Skyhook, http://www.skyhookwireless.com, 2013.
[10] G. Rivela, P. Scavini, D. Grasso, A. Calcagno, M.G. Castro, G. Di Chiara, G. Avellone, G. Cali', and S. Scaccianoce, "A 65 nm CMOS Low Power RF Front-End for L1/E1 GPD/Galileo Signals," *Proc. 21st Great Lakes Symp. VLSI (GLSVLSI),* pp. 217-222, 2011.
[11] ARM, "Big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7," http://www.arm.com/files/downloads/big_LITTLE_Final_Final.pdf, 2011.
[12] Mirasol, http://www.mirasoldisplays.com, 2013.
[13] S.H. Wu, C.M. Chen, and M.S. Chen, "An Asymmetric and Asynchronous Energy Conservation Protocol for Vehicular Networks," *IEEE Trans. Mobile Computing,* vol. 9, no. 1, pp. 98-111, Jan. 2010.
[14] X. Zhao, C. Hayner, M. Kung, and H. Kung, "In-Plane Vacancy-Enabled High-Power Si-Graphene Composite Electrode for Lithium-Ion Batteries," *Advanced Energy Materials,* vol. 1, no. 6, pp. 1079-1084, 2011.
[15] V. Vapnik, *Statistical Learning Theory.* J. Wiley & Sons, 1998.
[16] N. Samaan and A. Karmouch, "A Mobility Prediction Architecture Based on Contextual Knowledge and Spatial Conceptual Maps," *IEEE Trans. Mobile Computing,* vol. 4, no. 6, pp. 537-551, Nov./Dec. 2005.
[17] R. Azuma and G. Bishop, "A Frequency-Domain Analysis of Head-Motion Prediction," *Proc. ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAGH),* pp. 401-408, 1995.
[18] B. Liang and Z. Haas, "Predictive Distance-Based Mobility Management for Multidimensional PCS Networks," *IEEE/ACM Trans. Networking,* vol. 11, no. 5, pp. 718-732, Oct. 2003.
[19] J. Liang, C. Shaw, and M. Green, "On Temporal-Spatial Realism in the Virtual Reality Enviornment," *Proc. Fourth Ann. ACM Symp. User Interface Software and Technology (UIST),* pp. 19-25, 1991.
[20] S. Julier, J. Uhlman, and H.F. Durrant-White, "A New Method for Nonlinear Transformation of Means and Covariances in Filters and Estimators," *IEEE Trans. Automatic Control,* vol. 45, no. 3, pp. 477-482, Mar. 2000.
[21] M. Isard and A. Blake, "CONDENSATION - Conditional Density Propagation for Visual Tracking," *Int'l J. Computer Vision,* vol. 29, no. 1, pp. 5-28, 1998.
[22] A. Chan, R.W.H. Lau, and L. Li, "Hand Motion Prediction for Distributed Virtual Environments," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 1, pp. 149-150, Jan./Feb. 2008.
[23] Y. Tao, C. Papadias, and B. Lin, "Prediction and Indexing of Moving Objects with Unknown Motion Pattern," *Proc. ACM Int'l Conf. Management of Data (SIGMOD),* pp. 611-622, 2004.
[24] A. Quintero, "A User Pattern Learning Strategy for Managing Users' Mobility in UMTS Networks," *IEEE Trans. Mobile Computing,* vol. 4, no. 6, pp. 552-566, Nov./Dec. 2005.
[25] I. Akyildiz and W. Wang, "The Predictive User Mobility Profile Framework for Wireless Multimedia Networks," *IEEE/ACM Trans. Networking,* vol. 12, no. 6, pp. 1021-1035, Dec. 2004.
[26] D. Katsaros and Y. Manolopoulos, "Prediction in Wireless Networks by Markov Chains," *IEEE Wireless Comm.,* vol. 16, no. 2, pp. 56-63, Apr. 2009.
[27] J.K. Lee and J. Hou, "Modeling Steady-State and Transient Behaviors of User Mobility: Formulation, Analysis, and Application," *Proc. ACM MobiHoc,* pp. 85-96, 2006.
[28] A. Nicholson and B. Noble, "BreadCrumbs: Forecasting Mobile Connectivity," *Proc. ACM MobiCom,* pp. 46-57, 2008.
[29] H. Abu-Ghazaleh and A.S. Alfa, "Application of Mobility Prediction in Wireless Networks Using Markov Renewal Theory," *IEEE Trans. Vehicular Technology,* vol. 59, no. 2, pp. 788-802, Feb. 2010.
[30] A. Bhattacharya and S. Das, "LeZi-Update: An Information-Theoretic Framework for Personal Mobility Tracking in PCS Networks," *Wireless Networks,* vol. 8, nos. 2/3, pp. 121-135, 2002.
[31] L. Song, R. Jain, and X. He, "Evaluating Location Predictors with Extensive Wi-Fi Mobility Data," *Proc. IEEE INFOCOM,* pp. 1414-1424, 2004.
[32] K. Laasonen, "Cluster and Prediction of Mobile User Routes from Cellular Data," *Proc. Ninth European Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD),* pp. 569-576, 2005.
[33] G. Liu and G. Maguire Jr., "A Class of Mobile Prediction Algorithm for Wireless Mobile Computing and Communications," *Mobile Networks and Applications,* vol. 1, no. 2, pp. 113-121, 1996.
[34] T. Anagnostopoulos, C. Anagnostopoulos, S. Hadjiefthymiades, M. Kyriakakos, and A. Kalousis, "Predicting the Location of Mobile Users: A Machine Learning Approach," *Proc. ACM Int'l Conf. Pervasive Services (ICIS),* pp. 65-72, 2009.
[35] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A System for Learning Statistical Motion Patterns," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 9, pp. 1450-1464, Sept. 2006.
[36] D. Vsaquez, F. Large, T. Frichard, and C. Laugier, "Moving Obstacles' Motion Prediction for Autonomous Navigation," *Proc. Eighth Control, Automation, Robotics and Vision Conf. (ICARV),* pp. 149-154, 2004.
[37] J. Platt, "Probabilistic Outputs for Support Vector Machines and Comparison to Regularized Likelihood Methods," *Advances in Large Margin Classifiers,* A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, eds., pp. 61-74, MIT Press, 1999.
[38] T. Wu, C. Lin, and R.C. Weng, "Probability Estimates for Multi-Class Classification by Pairwise Coupling," *J. Machine Learning Research,* vol. 5, pp. 975-1005, 2004.
[39] M. Buhmann, *Radial Basis Functions: Theory and Implementations.* Cambridge Univ., 2003.
[40] Y. Linde, A. Buzo, and R.M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Comm.,* vol. 28, no. 1, pp. 84-95, Jan. 1980.
[41] M. Kim, D. Kotz, and S. Kim, "Extracting a Mobile Model from Real User Traces," *Proc. IEEE INFOCOM,* pp. 1-13, 2006.
[42] A. Mei and J. Stefa, "SWIM: A Simple Model to Generate Small Mobile Worlds," *Proc. IEEE INFOCOM,* pp. 2106-2113, 2009.
[43] I. Rhee, M. Shin, S. Hong, K. Lee, and S. Chong, "On the Levy-Walk Nature of Human Mobility," *Proc. IEEE INFOCOM,* pp. 1597-1605, 2008.

Addison Chan received the BEng degree in computer engineering from the Hong Kong University of Science and Technology and the MPhil degree in computer science from the City University of Hong Kong. He is currently working toward the PhD degree at Durham University in the United Kingdom. He has been participating in and leading a wide range of R&D projects in both academic and industrial sectors. Those projects include mobile computing, machine learning, location-based services, and multimedia. Some of his work has been published in renowned journals and conferences.

Frederick W.B. Li received the BA and MPhil degrees from Hong Kong Polytechnic University and the PhD degree from the City University of Hong Kong. He is currently a lecturer at Durham University. He was an assistant professor at Hong Kong Polytechnic University from 2003 to 2006, and the project manager of a Hong Kong Government ITF-funded project from 2001 to 2003. He serves as an associate editor of the *International Journal of Distance Education Technologies* and has served as a guest editor of two special issues of the *Journal of Multimedia*. He has served as a program cochair of ICWL 2007-2008 and IDET 2008-2009. His research interests include distributed virtual environments, computer graphics,and e-learning systems.